UDC: 004.93 https://doi.org/10.15407/jai2025.04.069

S. Kashperova[1], **N. Shapoval**[2]
[1,2]National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Ukraine
37 Beresteysky Avenue, Kyiv, 03056
[1]kashperova.study@gmail.com
[2]shovgun@gmail.com
[1]https://orcid.org/0009-0007-6442-7010
[2]https://orcid.org/0000-0002-8509-6886

ZERO--SHOT AND FEW-SHOT NAMED ENTITY RECOGNITION FOR THE UKRAINIAN LANGUAGE BASED ON A MODIFIED GLINER ARCHITECTURE

Abstract. This paper presents a method for named entity recognition (NER) for the Ukrainian language with limited training data, based on a modified GLiNER architecture. The proposed method combines the advantages of end-to-end span-based NER with architectural and training improvements added to handle low-resource issues. We replace the original DeBERTa encoder with a compact Snowflake Arctic-Embed 2.0 encoder pretrained for retrieval, introduce a post-fusion cross-attention block between text and entity type descriptions, and use a lightweight span scoring module with GoLU activation. A new Ukrainian multi-domain NER corpus and out of domain benchmark were created to evaluate the model. Experimental results show that the proposed method achieves competitive F1 performance compared to open-weight LLMs while being far cheaper to deploy, approaching proprietary LLM quality in few--shot settings.

Keywords: low--resource NLP tasks, named entity recognition, zero-shot learning, few--shot learning, retrieval.

Introduction

Named Entity Recognition (NER) is a core task in natural language processing that aims to identify spans in text that mention real-world entities and assign them to predefined classes. NER is a critical component of information extraction pipelines and is widely used to enrich unstructured documents with semantic metadata. In modern retrieval systems, recognized entities can be indexed as structured fields, used for entity-aware ranking, filtering, and query understanding, and they often serve as anchors for linking documents to canonical concepts in downstream applications such as Retrieval-Augmented Generation (RAG).

Beyond retrieval, NER plays an important role in semantic annotation and knowledge base construction. Extracted entities and their types enable the creation of knowledge graphs by supporting entity linking and relation extraction.

Traditionally, NER focuses on three major entity types — persons, locations, and organizations. But real industrial use cases frequently require a much broader inventory, including dates and time expressions, monetary amounts, percentages, products, events, and domain-specific categories. Expanding the label set makes text analytics more informative and useful for business applications, but it also increases task complexity and the amount of labeled data needed to train robust models. Moreover, errors made at the entity recognition stage tend to propagate to later components, potentially degrading analytics, decision-support systems, and leading to incorrect conclusions — making accurate and reliable NER a persistent practical challenge.

Modern large language models (LLMs) have demonstrated the ability to perform named entity recognition in zero--shot and few--shot setups thanks to prompting and instruction tuning. However, such solutions contain billions of parameters and require big computational resources both during training and inference, which limits their practical application in industrial projects with strict latency and cost constraints. For the Ukrainian language, an additional challenge is the limited availability of public NER- annotated corpora and the absence of standardized benchmarks for zero-shot and few-shot scenarios, in contrast to English- language resources such as Few-NERD [4].

Related work

Research on few-shot and zero-shot NER has progressed through several

architectural families, each attempting to solve the lack of annotated data. Early metric -learning approaches adapted prototypical networks [5] for NER by representing each entity type using a class prototype constructed from support examples. While simple and conceptually appealing, these methods struggle when entity span boundaries vary significantly or when entity types have complex semantics that cannot be represented by a single prototype vector. This limitation motivated the development of more sophisticated architectures such as CONTaiNER [6] and TadNER[7], which introduce contrastive and decomposed learning objectives to better capture the relationship between entity representations and type description

The CONTaiNER [6] architecture incorporates contrastive learning to increase type–span distinctions and force the model to produce discriminative embeddings under few--shot supervision. Although effective in benchmarking settings, CONTaiNER over-relies on contrastive losses and heavy augmentation strategies, making its performance sensitive to hyperparameter choices and batch composition. Moreover, CONTaiNER tightly couples the span detection and type discrimination processes, which can cause the model to generalize poorly when entity types differ significantly from those seen during training. So, in practice, CONTaiNER is less robust in low--resource or complex morphological languages such as Ukrainian.

The TadNER [7] framework attempts to address some of these issues by explicitly decomposing NER into two sub-tasks: span detection and type classification. However, this decomposition introduces new challenges. Span detection quality often becomes a bottleneck: errors in the first stage propagate directly to the classifier, and TadNER has limited mechanisms for recovering from span boundary mistakes. Additionally, TadNER relies heavily on supervised span annotations and is not naturally suited for zero--shot setup, since it assumes fixed entity types known at training time.

In contrast, GLiNER [1] adopts a fundamentally different viewpoint by treating NER as a joint span–type matching problem, where both spans and entity types are embedded into a shared semantic space. This formulation naturally accommodates zero--shot and open vocabulary scenarios, since the model does not rely on a fixed label set but instead uses textual descriptions of entity types. GLiNER also eliminates the need for prototype construction or type specific classifiers by predicting entity labels directly through similarity scoring.

Despite these advantages, GLiNER also has several issues. First, its performance depends heavily on the quality of the underlying encoder, and multilingual encoders often produce suboptimal representations for Ukrainian. Second, the interaction between entity type descriptions and text is weak, relying mostly on shared transformer layers rather than explicit cross--attention mechanisms. As a result, the model may misinterpret semantically similar entity types or fail to distinguish differences in type descriptions. Finally, GLiNER's scoring module is effectively a linear classifier, limiting its ability to model non-linear compatibility patterns between spans and types.

Statement of the problem

The objective of this work is to develop a zero--shot and few--shot NER method for the Ukrainian language that:

- provides a high level of generalization to new domains and entity types in the absence of, or with only minimal, annotated data;

- is computationally efficient and can be deployed on a single GPU with 16–24 GB of VRAM;

- achieves quality comparable to modern openweight LLMs in zero-shot and few-shot scenarios, while reducing inference cost.

To address the problem of named entity recognition for Ukrainian under limited annotated data, this paper tackles two closely related tasks: (1) the creation of high-quality annotated datasets, and (2) the design and evaluation of an effective NER method.

First, we construct a new Ukrainian corpus annotated with named entities, covering

texts from multiple domains. Since existing public resources are limited, this custom corpus is developed as part of the research.

Second, we develop a separate benchmark dataset to evaluate the model's ability to generalize to new domains. This benchmark makes it possible to evaluate how well the system recognizes entities in texts that differ from the training data and to systematically measure cross-domain robustness.

Together, the constructed resources and the proposed method form a unified experimental framework. The obtained results make it possible to compare the model with LLMs in terms of performance, cost, and generalization, thereby clarifying its practical value for Ukrainian NLP.

Theoretical foundations of zero--shot and few-shot NER

In zero--shot and few--shot NER, the model must assign entity labels when only a few or no annotated examples are available for the target types. Instead of learning a fixed BIO tag set, generalist approaches like GLiNER treat NER as a span–type matching problem: entity types are represented by natural -language descriptions, and the model predicts which spans in the text are compatible with which type descriptions.

GLiNER jointly encodes the input text and all entity type descriptions in a single transformer encoder. The input sequence is constructed by concatenating the document tokens with the list of entity type prompts, separated by special tokens, schematically:

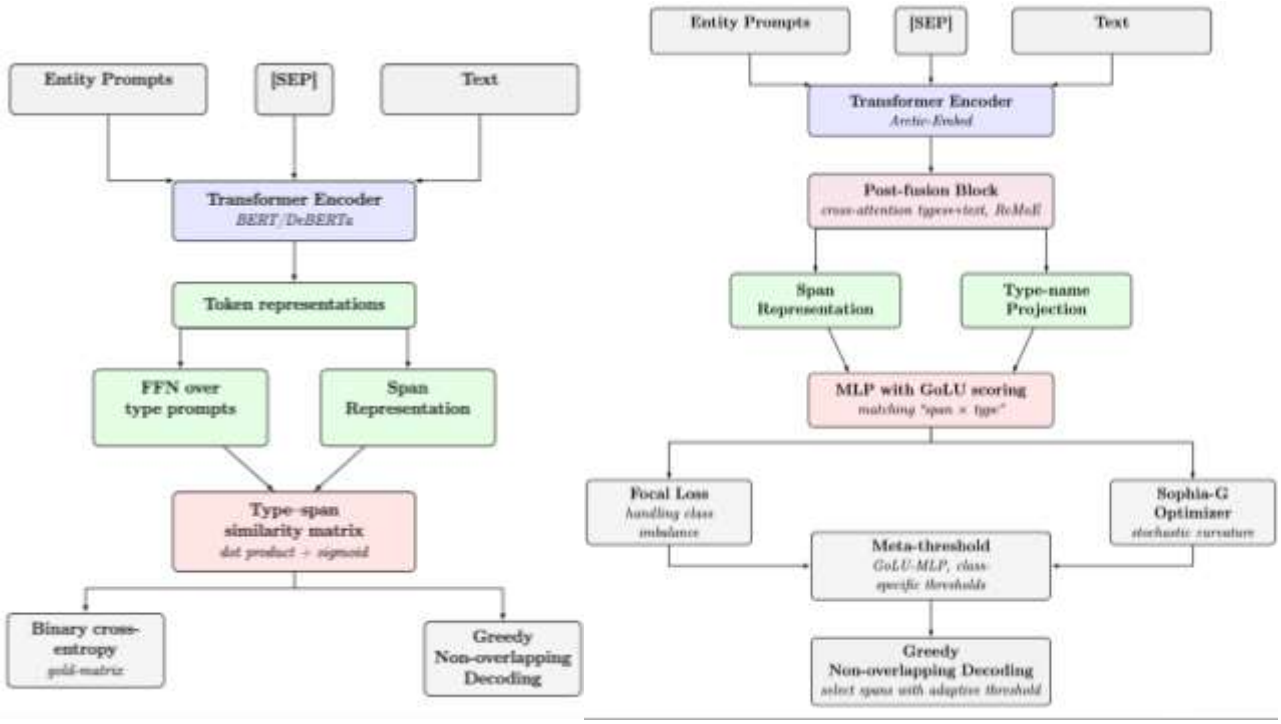


Fig. 1. Comparison of the GLiNER method (left) and the proposed ZeroNER method (right)

[CLS] x1 x2 ... xn [SEP]
[ENT] type1 description [SEP]
[ENT] type2 description [SEP]

Token representations from the encoder are pooled to form span vectors for all candidate spans and type vectors for each description starting at an [ENT] token. For each "span–type" pair, a scalar product is computed and a sigmoid function is applied to obtain the probability that the span belongs to the type.

During training, binary cross-entropy is minimized between the predicted and target type–span matrices.

In GLiNER, the decoding stage is separated from the scoring stage. It works as a greedy selection procedure with thresholding and prioritization. First, for each sentence, only the candidates with a score $(i, j, t) > \tau$

(typically τ = 0.5) are kept. These candidates are then pushed into a priority queue sorted by their scores in descending order. The model then selects the best candidates one by one.

In the flat NER setting, any overlap between spans is forbidden, so after choosing a span, all overlapping candidates are removed. In the nested setting, fully nested entities are allowed, but partial overlaps are not. Using a priority-queue-based decoding algorithm gives a complexity of $O(n \cdot log\, n)$, where n is the number of candidate span–type pairs. This approach scales well for long texts or large sets of entity types.

Proposed method

To improve performance under lowresource conditions and adapt NER to Ukranian, we propose the following key changes:

1. *Replacing the encoder with a compact Snowflake ArcticEmbed 2.0L encoder with matryoshka embeddings.* We replace the DeBERTa encoder with a multilingual encoder pretrained using a retrievaloriented RetroMAE scheme [8]. The Snowflake encoder contains 568 million parameters and produces vectors of size 1024 [2]. The model supports Matryoshka Representation Learning (MRL) [9], which allows it to reduce the vector size to 256 with only a 1.8% drop in quality (according to the MTEB Retrieval benchmark [10]).

2. *Postfusion crossattention block for types↔text.* We add a block after the main encoder in which type vectors and text token vectors interact via multihead crossattention. Types act as keys and values and text spans as queries, and vice versa, which improves alignment between the semantics of type descriptions and the text context.

3. *Lightweight span scoring module with GoLU.* Instead of a simple dot product, we use a small multilayer perceptron with GoLU activation [3] to compute the compatibility score for each "span–type" pair. This makes it possible to model more complex nonlinear dependencies without significantly increasing the number of parameters.

4. Loss function focusing on hard examples. We employ a modified focal loss [11] for the type–span matrix, which reduces the impact of the numerous negative examples and focuses learning on hard positive and misclassified spans.

5. *Contrastive term for structuring the representation space.* To better separate correct and incorrect span–type pairs, we add a small contrastive loss term that encourages the score of a true pair to be higher than the scores of negative types by a fixed margin. Formally, for each span *s* and its correct type *t*, the loss penalizes cases where a negative type t^- receives a score comparable to or higher than *t*. This helps the scoring module learn a clearer decision boundary and reduces confusion between semantically similar types.

Training corpus and benchmark for Ukrainian

For model training we combined a domain-rich news corpus with a more general corpus of grammatically correct sentences. The goal was to give the model both "hard" examples with dense entity mentions and long, multi-word names, and simpler sentences that help stabilize span boundaries and type generalization outside the news domain.

We used two open datasets. The first is a collection of Ukrainian journalistic and investigative texts on socio-political topics [12]. The second is the Ukrainian subset of the high quality multilingual sentences corpus[13], which contains diverse, grammatically well-formed sentences. Both datasets are available on the Hugging Face Datasets platform. Together they provide about 36000 training examples.

Primary annotation is produced with the GPT4o model [14], used as a NER extractor. The model receives an instruction in Ukrainian describing the target types, nesting rules and multityping, a strict JSON schema that forbids arbitrary keys, and a lowtemperaturewith retries and exponential backoff to improve determinism. On top of the LLM output we apply deterministic postprocessing: (1) boundary checking and automatic correction to the first exact match of the surface form, (2) type normalization (trimming, lowercasing, duplicate removal and order stabilization), and (3) dropping entries that fail validation (empty

type lists, incorrect offsets, spans not found in the text). The result is a consistent JSONL corpus suitable for training.

To study cross-domain robustness we also build an out-of-domain benchmark from five public Ukrainian datasets on Hugging Face: UAReviews [15], Wiki news -multilingual (uk subset) [16], UAL-topics [17], COSMUS [18] and LinguaNova (uk subset) [19]. These corpora cover user reviews, news, legal queries, Telegram comments and mixed web text, respectively.

From each dataset we sample 200 examples, giving a 1000 -sentence benchmark

Table 1. Out-of-domain benchmark datasets for Ukrainian NER

Dataset	*Domain*	*Samples*	*Role*
UAReviews[15]	Product and service reviews	200	Short, subjective texts
Wikinews [16]	News articles	200	Many ORG / GPE / event entities
UALtopics [17]	Legal questions	200	Domain-specific legal terminology
COSMUS [18]	Telegram posts and comments	200	Informal, noisy user text
LinguaNova[19]	Web texts	200	Mixed topics and styles
Total	—	1000	—

that covers different styles of Ukrainian. Initial entity labels are produced with GPT-4o [14] and then manually validated. For the few-shot setting we also build a small demonstration corpus: unique entity names are collected from the benchmark, and for each type a handful of additional examples are generated and checked, so that the same set of demonstrations can be used either in prompts or for few-shot tuning.

Experimental setup

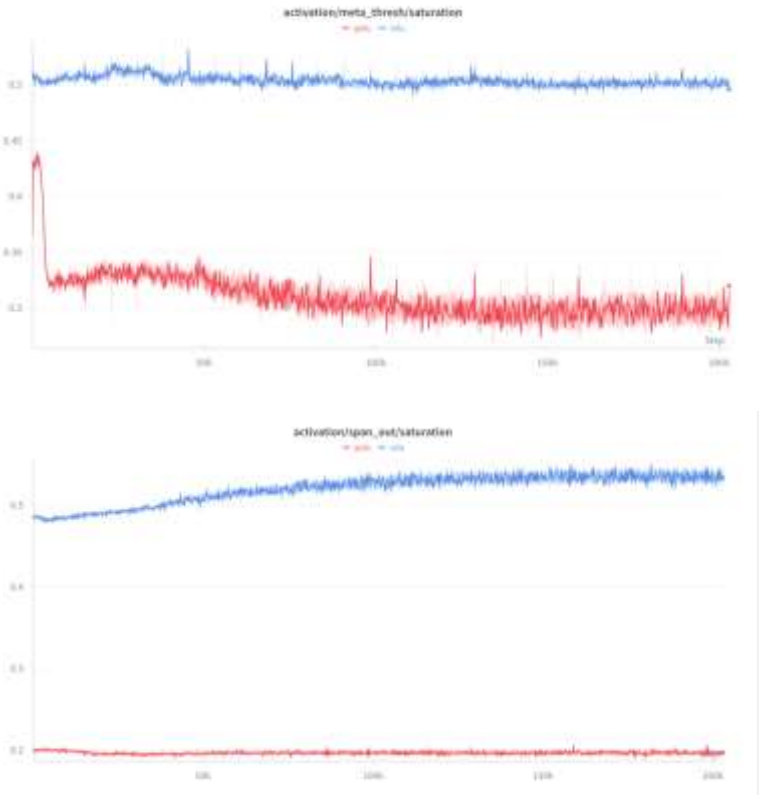
For training we used a cloud GPU provider instead of local hardware. For this work we selected the Modal Labs platform [20], which provides a Python-first API for defining infrastructure, fast container startup and serverless GPU execution. On Modal, an NVIDIA A10040GB instance costs about $0.000583 per second (≈$2.1 per hour) plus small per-second fees for CPU and RAM, and includes a monthly free credit for small projects [20].

All experiments were run on a single NVIDIA A100 with 40 GB of VRAM. To keep GPU memory usage under control we applied LoRA fine-tuning [21] instead of updating all encoder parameters. Low-rank adapters were attached to the query, key, value and output projections in the attention blocks of both the text encoder and the type-description encoder. Together with the post-fusion block, this setup trains roughly 10% of all parameters, while the base encoder weights remain frozen. We also used gradient accumulation to simulate a global batch size of 16 on hardware that can only hold a mini-batch of 8 examples, cosine learningrate scheduling with a short warmup phase, and adaptive gradient clipping with a gradually decreasing max_grad_norm.

The base encoder is Arctic-Embed-L v2.0, which originally has about 568M parameters and a large multilingual vocabulary [2]. Since we only need Ukrainian and English, we pruned the vocabulary and removed unused tokens, reducing the embedding matrix from approximately 256M parameters to about 106M. The final model has about 470M parameters in total (≈1.8 GB in FP32). Table 2 shows the distribution of parameters across major blocks. For evaluation we report span -level precision, recall and F1- score: a prediction is counted as correct only if both span boundaries and the entity type match the annotation. All metrics are micro-averaged over entities. We also carried out several controlled ablations. First, we compared ReLU and GoLU activations [3] in all projection MLPs and in the scoring MLP. During training we logged the fraction of saturated neurons (very small activation derivative) to study gradient flow. Second, we compared three optimizers: AdamW [23], AdaFactor [22] and Sophia-G [24]. Finally, we evaluated three variants of the scoring module: a dot-product baseline, an MLP without contrastive loss, and an MLP with an additional prototype-style contrastive term.

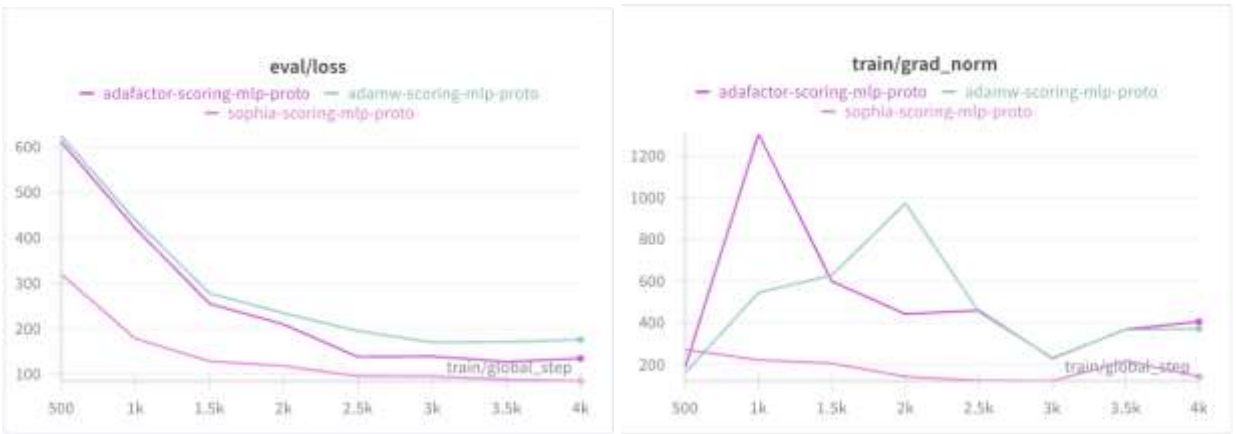
Table 2. Parameter distribution across model components after vocabulary pruning

Block	Number of parameters
Logit scale	1
Encoder	426734592
Postfusion block	29411340
Span representation module	7095296
Prompt representation module	2099712
Span head	1051136
Prompt head	1051136
Total	469546510



(a) Saturation dynamics for threshold block (b) Saturation dynamics for projection layers

Fig. 2. Comparison of activation saturation between GoLU and ReLU



(a) Validation loss function (b) Gradient norm dynamics

Fig. 3. Comparison of AdaFactor, AdamW and SophiaG optimizers

Results and discussion

In the activation study, GoLU showed much lower saturation than ReLU in the projection and span-output layers: for GoLU, only about 20% of neurons were saturated on average, while for ReLU this fraction was close to 50% (Fig.2). This suggests that GoLU keeps more neurons in an active regime and preserves gradient flow throughout training. In the MoE-like components the saturation curves for GoLU started higher but converged to levels comparable with ReLU, indicating more flexible use of capacity in the early training phase.

Among optimizers, Sophia-G consistently achieved the lowest validation loss and the highest F1-score, while keeping gradient norms stable and reducing the need for aggressive clipping. AdamW and AdaFactor produced noisier gradient norms and slightly worse validation metrics (Fig. 3). Table 3 summarizes the main results.

Table 3. Validation metrics for different optimizers

Optimizer	Precision	Recall
SophiaG	0.72	0.87
AdaFactor	0.66	0.82
AdamW	0.64	0.79

Table 4. Comparison of scoring module variants

Configuration	Precision	Recall	F1score
MLP with proto	0.72	0.87	0.78
Dot product with proto	0.70	0.85	0.77
MLP without proto	0.68	0.83	0.75

Replacing the simple dot-product scoring with an MLP over [s, t, st, |s − t|] improved F1 -score, and adding a contrastive prototype-style loss gave the best overall results. As shown in Table 4, the MLP with contrastive loss achieved the highest precision and recall, indicating that the richer scoring module can better exploit span–type interactions.

We evaluated the proposed method on the five-domain benchmark described earlier and compared it with multilingual GLiNER [25], several open-weight LLMs (Qwen3-8B [26], Llama 3.1-8B [27], Gemma 3-12B-IT [28], GPT-OSS-20B [29] accessed via NovitaAI [30]) and two Ukrainian LLMs (MamayLM [31] and LappaLLM [32] running locally with vLLM [33]). We also included two proprietary LLMs, GPT-5-mini [34] and Haiku 4.5 [35], for reference.

Tables 5 and 6 show the F1 - scores over the five domains in zero-shot and few-shot scenarious. In zero-shot, the proposed model out-performs multilingual GLiNER and most small open LLMs, and is competitive with the larger GPT-OSS model, while remaining much smaller and cheaper to deploy. GPT-5-mini gives the best zero-shot F1 but requires a commercial API.

To better understand model behavior, we also inspected JSON outputs for complex sentences with multiple entities and distractor labels. The model performs very well on generic classes that do not require deep domain knowledge, and correctly handles labels. In specialized domains (e.g. finance) it occasionally overgeneralizes semantic similarity, assigning a broader class to spans that are semantically close in the encoder space. Typical examples include financial phrases where both true fund characteristics and related but more generic financial terms receive the same label.

Table 5. Zero-shot NER results (F1-score) for different models and datasets

Model	Reviews	Wiki	Law Q&A	Telegram	CC100	Avg F1
Proposed method	0.7317	0.8431	0.6863	0.6832	0.7141	0.7317
GLINERmultiv2.1	0.6776	0.7899	0.5740	0.6066	0.6262	0.6549
Qwen8B	0.7569	0.8172	0.6364	0.7380	0.5431	0.6983
Gemma12B	0.6888	0.8231	0.5617	0.7194	0.4894	0.6565
GPTOSS20B	0.7625	0.8288	0.6931	0.7460	0.5946	0.7250
Llama3.18B	0.5604	0.7397	0.5052	0.6172	0.4263	0.5698
LappaLLM	0.5781	0.7739	0.5244	0.5686	0.4731	0.5837
MamayLM	0.6413	0.7452	0.5958	0.6595	0.5109	0.6310
GPT5mini	0.6849	0.8327	0.7343	0.8118	0.6658	0.7459
Haiku 4.5	0.6749	0.7857	0.6064	0.6989	0.5807	0.6693

Table 6. Few-shot NER results (F1-score) for different models and datasets

Model	Reviews	Wiki	Law Q&A	Telegram	CC100	Avg F1
Proposed method	0.8123	0.8741	0.7432	0.7215	0.6500	0.7602
Qwen8B	0.7401	0.8204	0.6273	0.7125	0.6621	0.7125
Gemma12B	0.6932	0.8129	0.5581	0.6713	0.5650	0.6601
GPTOSS20B	0.7954	0.8510	0.7298	0.7810	0.6602	0.7634
Llama3.18B	0.5354	0.7388	0.4962	0.5981	0.4550	0.5647
LappaLLM	0.6021	0.7705	0.5510	0.6004	0.5460	0.5940
MamayLM	0.6584	0.7488	0.6031	0.6820	0.4971	0.6379
GPT5mini	0.7952	0.8649	0.7413	0.8200	0.6890	0.7821
Haiku 4.5	0.6920	0.7805	0.6082	0.6990	0.6853	0.7130

Conclusions

This work presented a compact zero- and few-shot NER model for the Ukrainian language that achieves competitive quality while remaining practical for real-world deployment. Unlike general-purpose LLMs, the proposed architecture does not rely on autoregressive decoding and does not require high-end hardware. On a standard Apple M2 CPU, the model produces predictions in approximately 300–500 ms per input example, making it suitable for local scenarios where GPU access is limited.

Experimental comparisons demonstrate that modern small and medium LLMs often struggle with structured NER output. These models frequently return invalid JSON objects, hallucinate additional entities, or rewrite text spans (for example converting an adjective into its noun form). Such behaviour breaks span alignment and significantly reduces the reliability of NER extraction, especially in zero-shot mode. In contrast, the proposed method operates on span–type scoring and produces deterministic, structure-preserving outputs without relying on generative decoding, which results in higher generalisation across domains.

The model also benefits from architectural modifications such as LoRA-based fine-tuning, vocabulary reduction, and a contrastive scoring term that improves the separation between compatible and incompatible span–type pairs. Together, these techniques allow the system to outperform GLiNER and several open-weight LLMs.

Future work will focus on several directions. First, extending the method to support richer hierarchical ontologies and multilevel entity typing. Second, integrating domain adaptive pretraining to improve performance in specialised domains such as law or finance. Finally, exploring alternative contrastive objectives and calibration mechanisms for confidence scoring.

Overall, the presented approach demonstrates that good-quality NER for Ukrainian can be achieved without large-scale generative models, offering an alternative suited for both research and practical applications.

References

1. Gubarev, V., Kuratov, Y., Dale, D., et al. (2024). GLiNER: Generalist and Lightweight Model for Named Entity Recognition. *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics*, 5364–5376. doi:10.18653/v1/2024.naacllong.300.

2. Snowflake Inc. (2024). Snowflake ArcticEmbedL v2.0: Multilingual text embedding model. Model card on Hugging Face Hub. Available at: https://huggingface.co/Snowflake/snowflakearcticembedlv2.0.

3. Enevoldsen, K., Zhang, Y., Stosic, A., et al. (2025). Gompertz Linear Units: Leveraging Asymmetry for Enhanced Learning Dynamics. *arXiv preprint*, arXiv:2502.03654. doi:10.48550/arXiv.2502.03654.

4. Ding, N., Xu, G., Chen, Y., Wang, X., Han, X., Xie, P., Zheng, H.T., Liu, Z. (2021). FewNERD: A Fewshot Named Entity Recognition Dataset. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, 3198–3213. doi:10.18653/v1/2021.acllong.248.

5. Snell, J., Swersky, K., Zemel, R. (2017). Prototypical Networks for FewShot Learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 30. doi:10.48550/arXiv.1703.05175.

6. Das, S. S., Katiyar, A., Passonneau, R. J., Zhang, R. (2022). CONTaiNER: Fewshot Named Entity Recognition via Contrastive Learning. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 6338–6353. doi:10.18653/v1/2022.acllong.439.

7. Li, Y., Yu, Y., Qian, T. (2023). TypeAware Decomposed Framework for FewShot Named Entity Recognition. *Findings of the Association for Computational Linguistics: EMNLP 2023*. doi:10.18653/v1/2023.findingsemnlp.598.

8. Xiao, S., Liu, Z., Shao, Y., Cao, Z. (2022). RetroMAE: PreTraining Retrievaloriented Language Models via Masked AutoEncoder. *Proceedings of EMNLP 2022*. doi:10.48550/arXiv.2205.12035.

9. Kusupati, A., Bhatt, G., Rege, A., Wallingford, M., Sinha, A., Ramanujan, V., ... & Farhadi, A. (2022). Matryoshka representation learning. *Advances in Neural Information Processing Systems*, *35*, 30233-30249. doi:10.48550/arXiv.2205.13147.

10. Muennighoff, N., Tazi, N., Magne, L., & Reimers, N. (2023, May). Mteb: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics* (pp. 2014-2037). doi:10.48550/arXiv.2210.07316.

11. Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision* (pp. 2980-2988). doi:10.1109/ICCV.2017.324.

12. Vitaliias. (2024). hromadske_corruption [dataset]. Hugging Face Datasets. url:https://huggingface.co/datasets/Vitaliias/hromadske_corruption, 2025. Accessed: 2025-10-26.

13. agentlans. (2023). Highquality multilingual sentences (subset: uk) [dataset]. Hugging Face Datasets. url:https://huggingface.co/datasets/agentlans/highqualitymultilingualsentences/viewer/uk, 2025. Accessed: 2025-10-26.

14. OpenAI. (2024). GPT4o models. url: https://platform.openai.com, 2025. Accessed: 2025-10-26.

15. KSE-RESEARCH-Group. (2023). UAReviews [dataset]. Hugging Face. url:https://huggingface.co/datasets/KSERESEARCHGroup/UAReviews, 2025. Accessed: 2025-10-26.

16. Fumika. (2023). Wiki news multilingual [dataset]. Hugging Face. url:https://huggingface.co/datasets/Fumika/Wikinewsmultilingual, 2025. Accessed: 2025-10-26.

17. Yehor. (2023). UALtopics [dataset]. Hugging Face. url:https://huggingface.co/datasets/Yehor/ualtopics, 2025. Accessed: 2025-10-26.

18. Shynkarov, Y. (2023). COSMUS [dataset]. Hugging Face. url:https://huggingface.co/datasets/YShynkarov/COSMUS, 2025. Accessed: 2025-10-26.

19. agentlans. (2023). LinguaNova [dataset]. Hugging Face. url:https://huggingface.co/datasets/agentlans/LinguaNov, 2025. Accessed: 2025-10-26.

20. Modal Labs. (2025). Highperformance serverless GPU platform. url: https://modal.com/, 2025. Accessed: 2025-10-26.

21. Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W. (2022). Lora: Low-rank adaptation of large language models. *ICLR*, *1*(2), 3. doi:10.48550/arXiv.2106.09685.

22. Shazeer, N., & Stern, M. (2018, July). Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning* (pp. 4596-4604). PMLR. doi:10.48550/arXiv.1804.04235.

23. Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*. doi:10.48550/arXiv.1711.05101.

24. Liu, H., Li, Z., Hall, D., Liang, P., & Ma, T. (2023). Sophia: A scalable stochastic second-order optimizer for language model pre-training. *arXiv preprint arXiv:2305.14342*. doi:10.48550/arXiv.2305.14342.

25. Urchade. (2024). glinermultiv2.1 [model]. Hugging Face. url: https://huggingface.co/urchade/gliner_multiv2.1, 2025. Accessed: 2025-10-26.

26. Qwen Team. (2024). Qwen38B [model]. Hugging Face. url:https://huggingface.co/Qwen/Qwen38B, 2025. Accessed: 2025-10-26.

27. Meta AI. (2024). Llama 3.18B [model]. Hugging Face. url:https://huggingface.co/metallama/Llama3.18B, 2025. Accessed: 2025-10-26.

28. Google DeepMind. (2024). Gemma 312BIT [model]. Hugging Face. url:https://huggingface.co/google/gemma312bi, 2025. Accessed: 2025-10-26.

29. OpenAI. (2024). GPTOSS20B [model]. Hugging Face. url: https://huggingface.co/openai/gptoss20b, 2025. Accessed: 2025-10-26.

30. NovitaAI. (2025). Model serving platform. url: https://novita.ai, 2025. Accessed: 2025-10-26.

31. INSAIT Institute. (2024). MamayLM Gemma312BIT v1.0 [model]. Hugging Face. url: https://huggingface.co/INSAITInstitute/MamayLMGemma312BITv1.0, 2025. Accessed: 2025-10-26.

32. Lapa Lab. (2024). LappaLLM [model]. Hugging Face. url:https://huggingface.co/lapallm/lapav0.1.2instruct, 2025. Accessed: 2025-10-26.

33. vLLM Team. (2024). vLLM: Easy, Fast, and Cheap LLM Serving. url: https://docs.vllm.ai/, 2025. Accessed: 2025-10-26.

34. OpenAI. (2025). GPT5mini models. url: https://platform.openai.com/docs/models/gpt5mini, 2025. Accessed: 2025-10-26.

35. Anthropic. (2025). Claude Haiku 4.5 models. url: https://www.anthropic.com/claude, 2025. Accessed: 2025-10-26.

The article has been sent to the editors 15.12.25.
After processing 17.12.25.
Submitted for printing 30.12.25.

Copyright under license CCBY-SA4.0.